Multiple Platforms: Porting Agent-Based Simulation from Grids to Graphics cards

Dr. Mariam Kiran

University of Bradford

Presented at:
Workshop on Portability Among HPC Architectures for Scientific Applications
Super Computing 2015

Agenda for the Talk

Introduction

Setting the stage: Agent-based modelling

Introducing FLAME and its portability from HPC to GPU

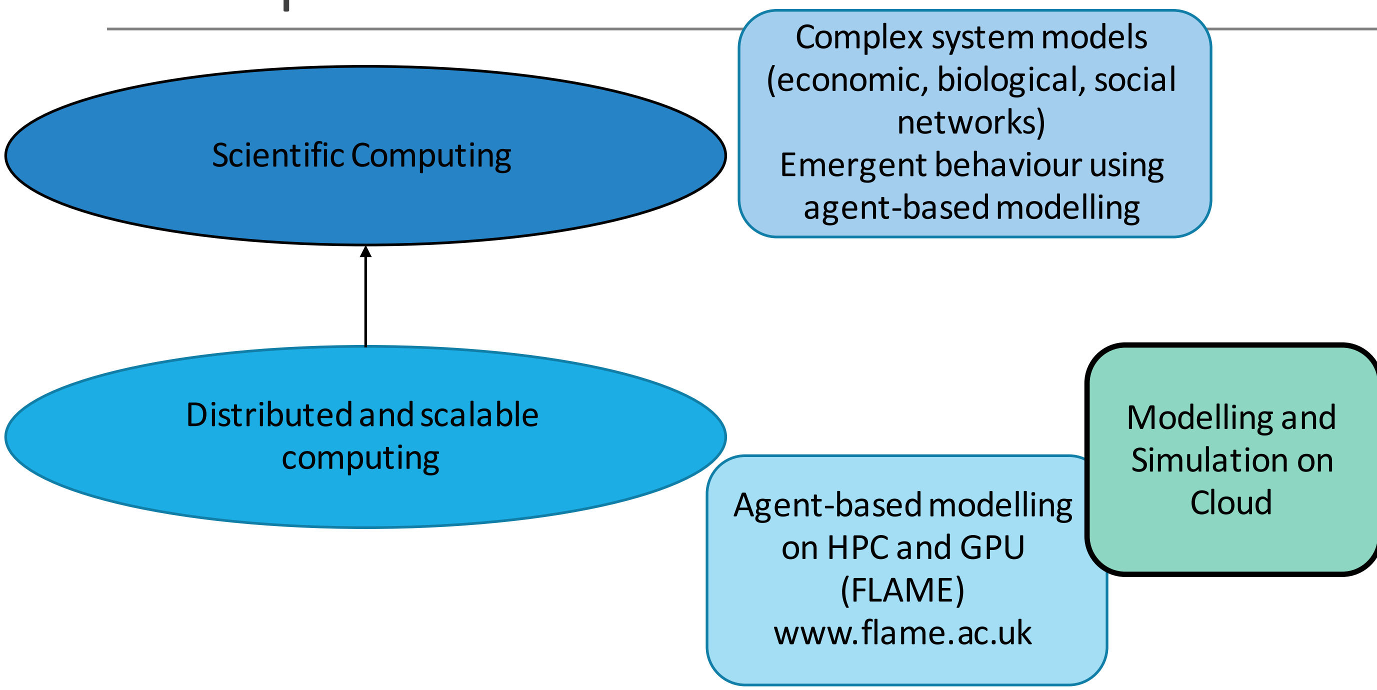
Portability problems – rewriting models to make sure they 'run'

Possible future directions of using Clouds

Research challenges

Conclusions

Computing working with other disciplines



Agent-based Modelling

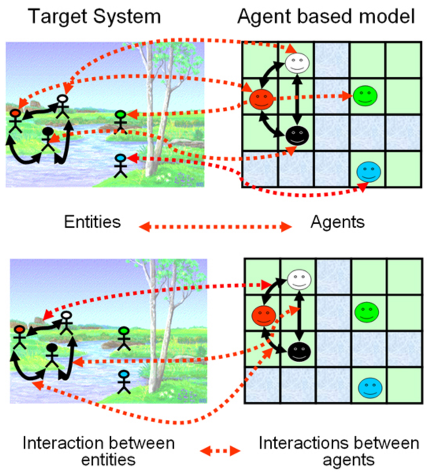
Each individual is a packet of programming code allowed to simulate together

Granularity

Interactions

Overcome most assumptions in the model

Examples in nature – bird flocking behaviour (boids), crowd behaviour in humans



Introducing FLAME for building agent-based models

Produced at the University of Sheffield, UK.

Flexible Large-scale Agent-based Modelling environment.

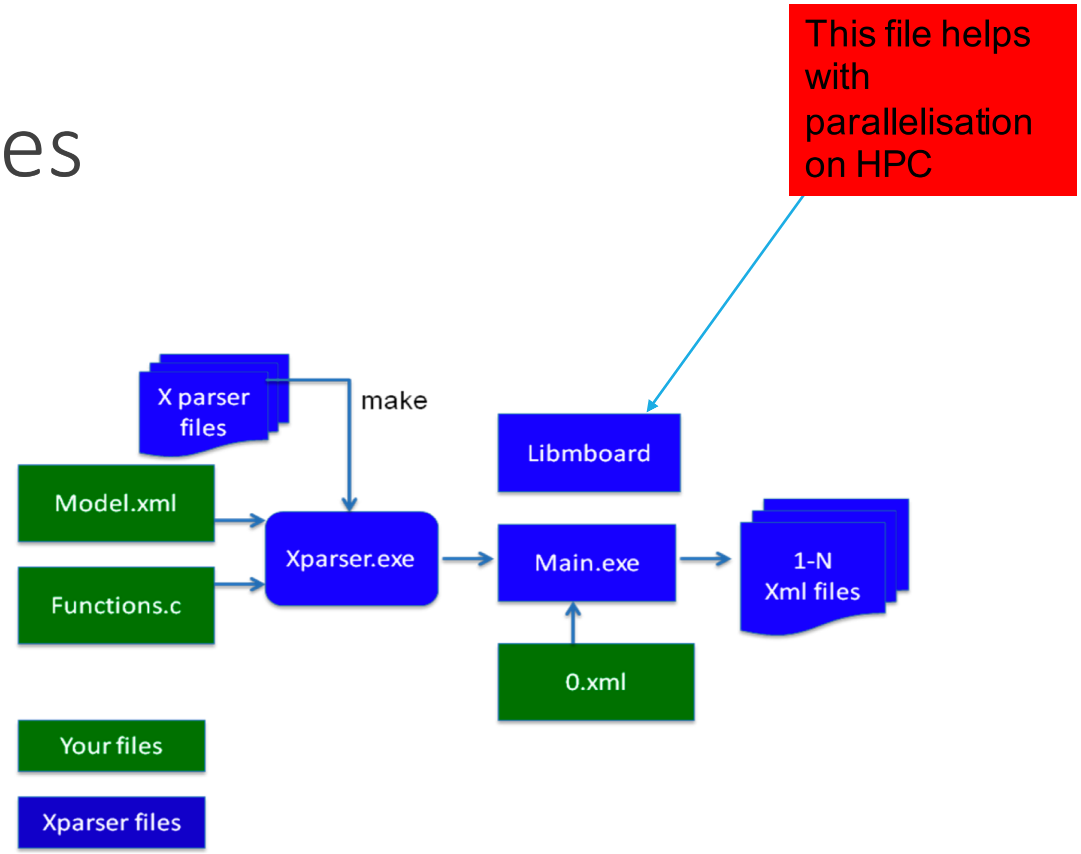
Based on X-machine architecture for agents.

Being used in a wide number of projects (Modelling of cells, tissues, biological and economic scenarios or networking models)

Automatically produces parallelisable code for models.

Ease of programming for non-computer experts.

FLAME files


This file helps with parallelisation on HPC
X parser files
make
Libmboard
Model.xml
Xparser.exe
Main.exe
1-N Xml files
Functions.c
0.xml
Your files
Xparser files

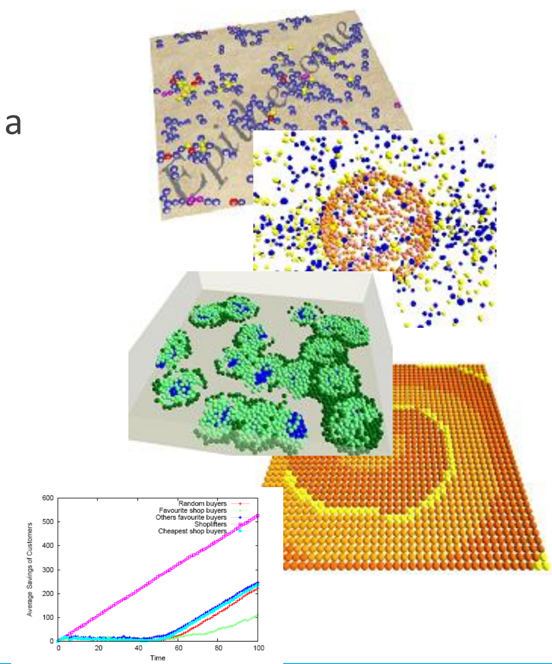
FLAME advantages

Produces automatically parallelisable code.

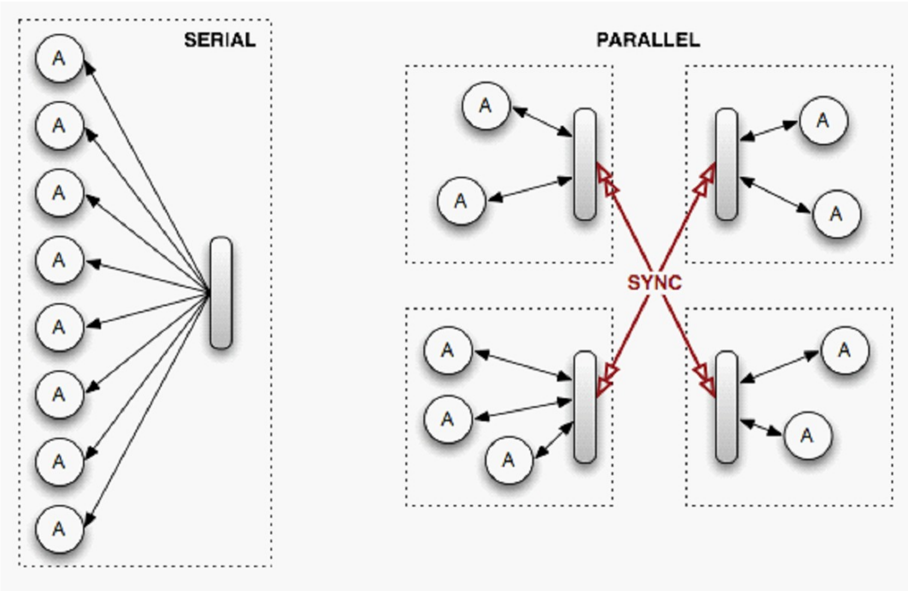
Default it runs in serial.

But by just adding a flag –p , while compiling, it produces a parallelisable model code.

Communication is handled using an intelligent message board library which can pool out relevant messages, sort, randomise or filter them on any criteria.



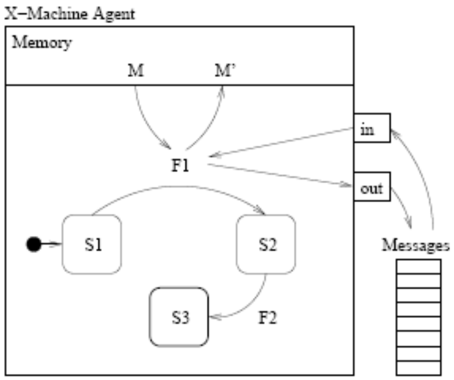
Libmboard – FLAME message board library



Software Engineering perspective…

Flame follows a strict X-machine architecture for its agents allowing it to contain

- Memory
- Functions
- States
- Messages in and out



Porting models:
from HPC to Graphics Cards

Multi platform capability

From FLAME

You can write models easily

Software works out how to distribute

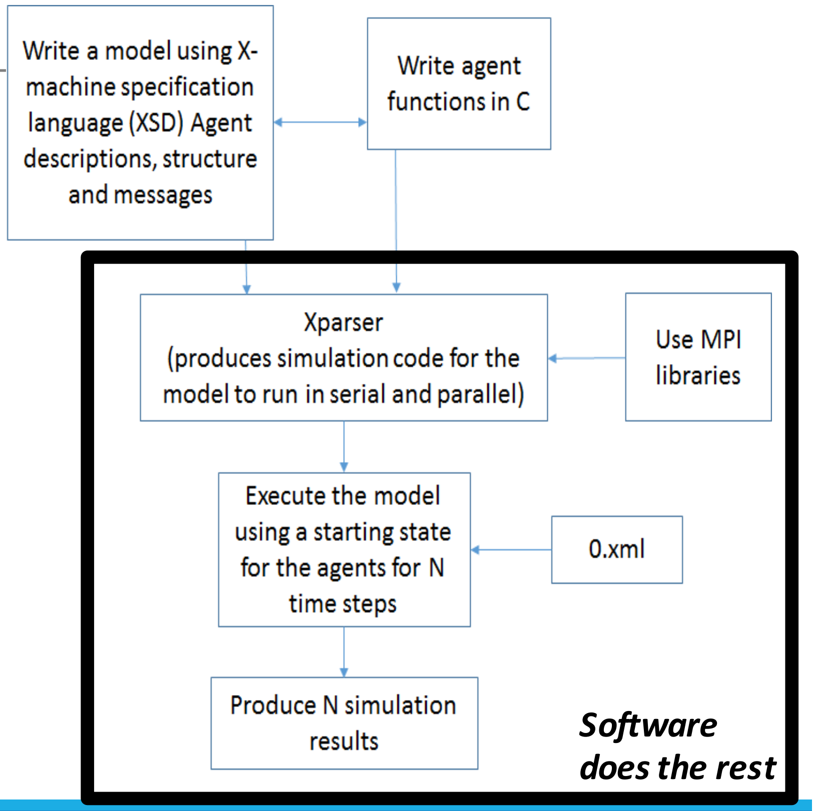
All you have to do is run

Need some knowledge of the system

Pre allocation and memory requirements

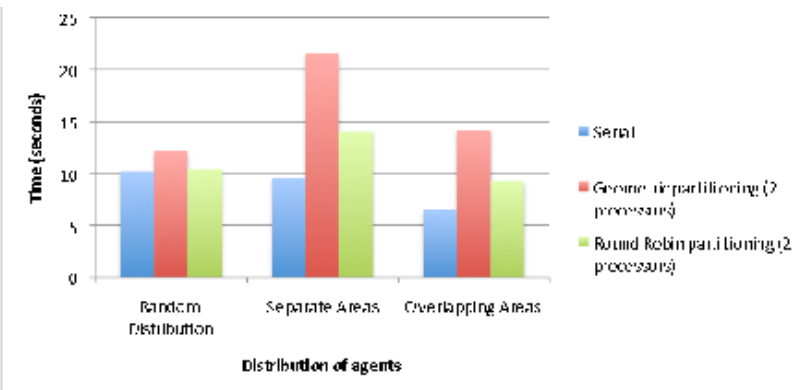
Sometimes you need to rewrite the complete model

How to test the model rewritten in correct?

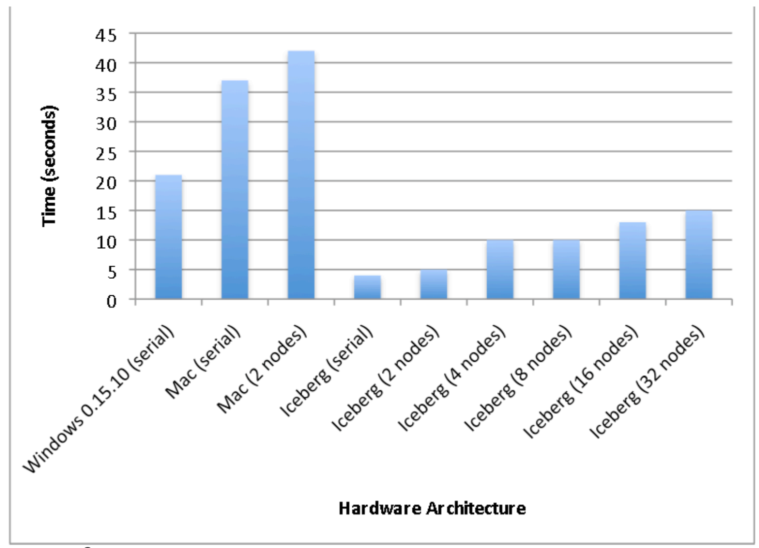


Same models running on (a) different distributions and (b) number of nodes

Changing placement of processes on processors- same model



Changing number of processors- same model



<u>Note:</u> Future work on Clouds may introduce a multitude of more factors – Energy, Cost, Optimal VM placement, networks, and many more

What are the performance characteristics

Main performance was Time

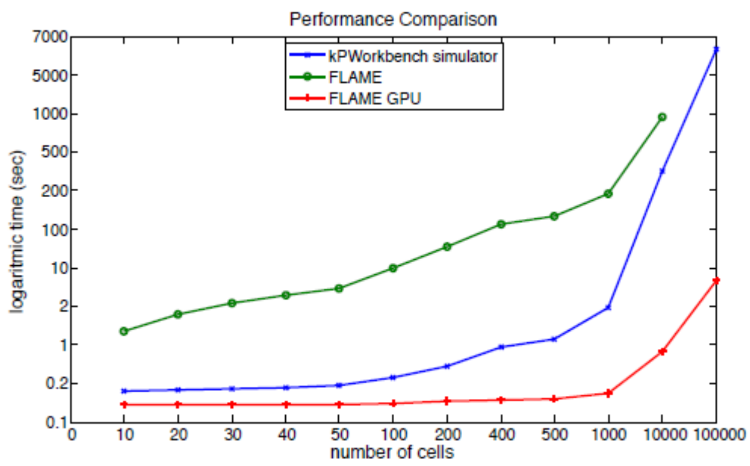
Do we need any more?

-Model checking

-Costs

-Eco-efficiency

-Data patterns – verifying results



HPC versus GPU

Considerable changes in the software to ensure it executes on GPU:

- Writing the Agents – different model descriptions
- Pre-allocation of Agent memory – GPU needs all advance knowledge – no dynamic allocation of memory is possible
- Message communication: need to break down bigger functions into simpler one message funtions
- Simpler versus Complex – Remove dynamic arrays- memory needs to be simple
- Looping through Messages – Loop traversal is different , rewrite functions
- Agent birth and death – Advance memory allocation
- Real time visualization – HPC does not allow this

Looping through Messages

Code for HPC needs to introduce a flag ‘finished’ to leave the while loop.

Else simulation hangs!

HPC implementation:

```
int MyFunction (xmahine_memory* agent,
xmachine_message_list* list) {
    xmachine_message* message = get_first_message(list);
    while(message) {
        if (message->id == agent->id) {
            agent->state += message->state;
            return 0;
        }
    message = get_next_message(message, list);
    }
    return 0;
}
```

GPU implementation:

```
int MyFunction(xmahine_memory* agent, xmachine_message_list* list) {
    bool finished = false;
    xmachine_message* message = get_first_message(list);
    while(message) {
        if (!finished) {
            if (message->id == agent->id) {
                agent->state += message->state;
                finished = true;
            }
        }
    message = get_next_message(message, list);
    }
    return 0;
}
```

What about cloud?

OPEN LABS

GIVING ACCESS EVERYWHERE

Computationally - why move towards Cloud?

Issues	**High performance computing**	**Cloud computing**
Kind of models and processing	Processing is limited is some architectures	Can introduce dynamic scalability for more complex processing.
Cost	Access to expensive hardware to model and simulate systems.	Resources can be hired as needed.
Failure recovery	No fault recovery when disk space runs out.	Applications can burst to more Clouds if needed, automatically.
Dynamic changes in the model	No real time processing, jobs are submitted to a queue, which means real time changes cannot be incorporated in the models.	Can execute jobs on the fly which can read real time data feeding to the models directly. (very useful for sensor related models)

Cloud Computational challenges relevant to ABM

Memory constraints saved as big data sets

Service level agreements

Optimise computation

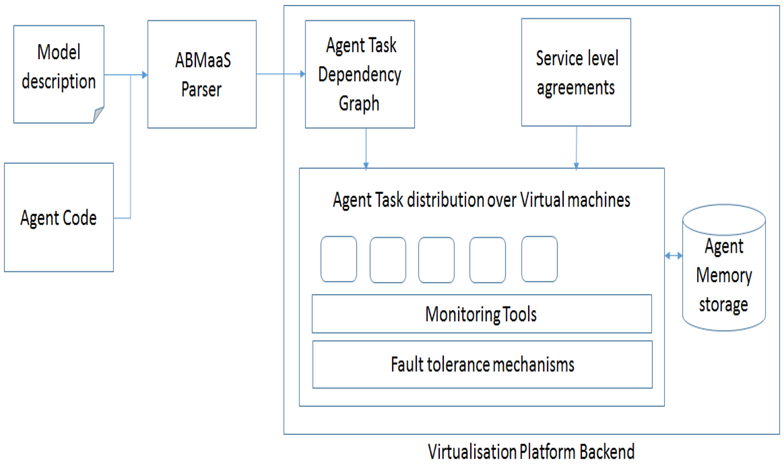
Reduce time

Reduce costs

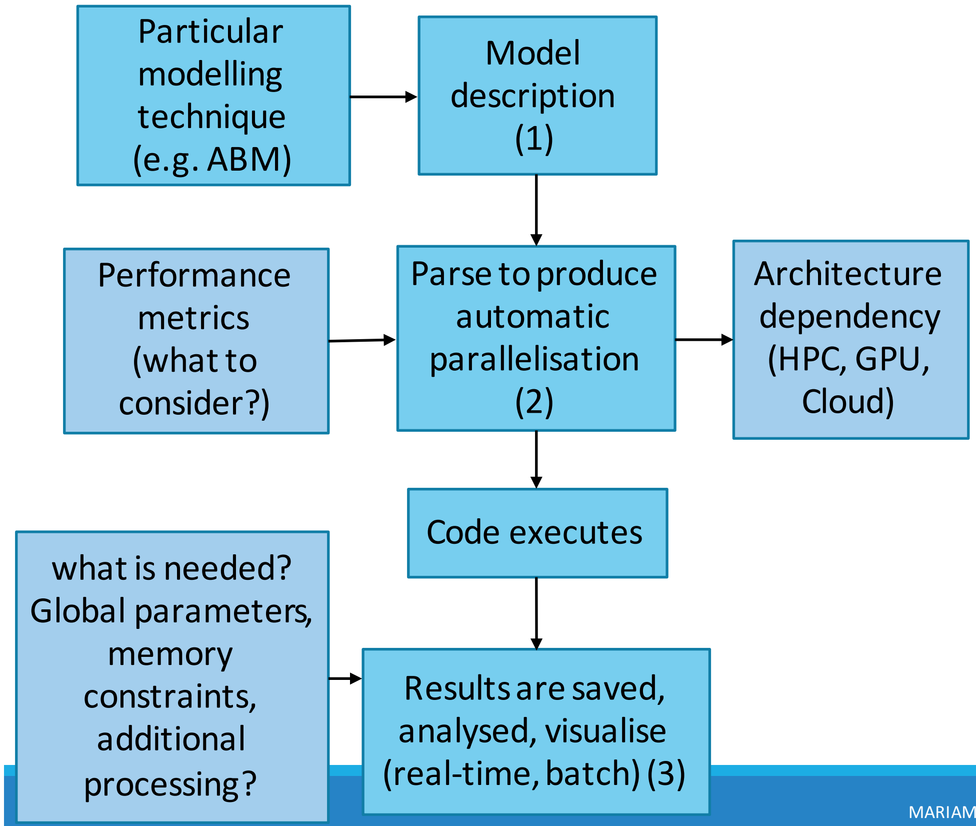
Energy efficient code

Hide away as much computation as possible

- VM doing processing
- Data being saved in data servers



Research challenges still relevant



1. Language to communicate models between multiple groups

2. Automatically parse this to optimal distribution for processing
   - Rewrite models for GPU, HPC, Cloud?
   - What are the performance metrics needed by modellers?

3. Just saving information needed, global averages, specific events for verification

‘Open lab’ to execute and share models and results?

Conclusions

Simulations are getting larger and more complex.

Realistic simulations require larger populations, or multiple types of population, as the validity of emergent characteristic dependent on both: the accuracy of the behaviour modelled and population sizes.

Forecast behaviours of systems faster than the wall-clock time.

Run-time costs are presently inhibiting the effective use of ABM as forecasting tool.

Agent-based models have successfully been able to uncover new aspects of economic systems such as the effect of migration on EU labor markets or uncovering some underlying facts in biological systems and need HPC, GPUs and newer technologies.

Many models written for HPC and GPU should portray similar characteristics, hiding away much of the software complexity from the non computing scientists using the tools to write their models which is a challenge in its own right.

Thankyou and Any questions?